## REMARKS

Applicant's remarks are provided in the order in which Examiner's various objections and rejections appear in the July 8, 2005, Office Action. Claims 1-27 are pending in the above-identified patent application. Claims 1, 10 and 19 are independent claims.

Specification

The examiner objected to the title as not being descriptive of the invention.

While the applicant believes that the current title complies with rule 37 C.F.R. §1.72(a), in the interest of expediting prosecution applicant has amended the title. The applicant thanks the examiner for the various proposed replacement titles, but believes that the amendment to the title as provided above more adequately and concisely indicates the nature of the subject matter disclosed in applicant's claims.

The applicant also amended for greater clarity the specification to remove part of the paragraph beginning at page 11, line 11.

The examiner objected to claims 4, 6, 13, 15, 22, and 24 on the ground that the wording "the first operand" lacks an antecedent basis. Applicant amended these claims to remove the word "first". Applicant notes that the independent claims from which claims 4, 6, 13, 15, 22, and 24 depend recite "an operand", thereby forming the antecedent basis for the amended wording "the operand".

The examiner also objected to claims 9, 8, and 27 on the ground that the wording "the result" lacks an antecedent basis. Applicant amended these claims to clarify that the result is "a result formed in the ALU", thereby traversing the examiner's objection. Support for this amendment may be found, for example, on page 12, lines 23-24 of the originally filed application.

The examiner further stated that if claims 5, 14 and 23 are found allowable, then claims 8, 17, and 26 would be objected to as being duplicates of claims 5, 14, and 23 respectively.

Applicant amended claims 17 and 26 to replace the word "shift" with the word "rotate".

Claim 8 already recites "rotate" and thus was not amended. Applicant thanks the examiner for pointing out the erroneous duplication that occurred in claims 17 and 26.

The examiner rejected claims 3-8, 12-17 and 21-26 under 35 U.S.C. §112, first paragraph, as failing to comply with the enablement requirement. Applicant amended claims 3-8 to provide that the instruction further comprises a field that indicates the various shifting and rotating operations to be performed. Claims 12-17 and 21-26 were similarly amended. Support for this amendment is found on page 11, of the originally filed application, describing the "opt_shf_cntl" field that represents shift or rotate of the register contents using a syntax shown in the table on that page

The examiner rejected claims 1-8, 10-17 and 19-26 under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 4,569,016 to Hao in view of the reference "Computer Organization and Design" by Hennessey and Patterson, and further in view of the reference "Computer Architecture, a Quantitative Approach", also by Hennessey and Patterson. The examiner has also rejected claims 9, 18, and 27 under 35 U.S.C. §103(a) as being unpatentable over Hao in view of "Computer Organization and Design" to Hennessy et al., and further in view of U.S. Patent No. 5,832,258 to Kiuchi.

Applicant has amended independent claim 1 to recite that the instruction that causes the ALU to load shifted values of a source operand into the destination register does so by selectively loading the data into any specified combination of bytes of the destination register. Support for the amended feature may be found, example, at page 11, lines 7-12 of the originally filed application. Applicant similarly amended independent claims 10 and 19. Applicant further amended claims 2, 11, and 20 to replace the wording "one or more bytes" with "bytes" thereby making the language of claims 2, 11, 20 consistent with the language of claims 1, 10, and 19, respectively.

Applicant's amended independent claim 1 requires one of the instructions to cause the ALU to "selectively load any specified combination of bytes of data within a transfer register associated with one of the plurality of microengines with a shifted value of an operand that preserves the bytes of data that are not loaded." Applicant's recited instruction specifies a

particular combination of bytes in the destination register into which shifted values of the source operand are loaded. The non-specified bytes are not loaded. For example, for a destination register having four (4) bytes, bytes 1 and 3 may be specified as the bytes into which the shifted values of the source operand is to be loaded, while bytes 0 and 2 remain unchanged after execution of the instruction.

In contrast, Hao discloses a set of instructions that utilize the functions of rotation, shifting and merging under a mask, and a mechanism for performing the same in a single machine cycle (abstract). For example, Hao explains, with reference to its rotate instruction, that "the result of the rotate instruction is either inserted into the register under control of the mask provided, or is AND'ed with the mask before being loaded into the register" "(Hao's col. 13, lines 1-4). The mask Hao uses is an 11-bit field in the instruction that indicates how to construct the mask that is to be used to AND the rotated operand into the register. The resultant constructed mask always includes a central string of consecutive 1's or 0's, surrounded by 0's or 1's respectively. Bit 21 of the mask field in the instruction indicates whether the central string is that of 1's or 0's. The next 5 bits in the mask field indicate the left most position of the central string in a 32-bit register, while the last 5 bits of the mask field indicate the right most position of the central mask string in the a 32-bit register (col. 12, lines 7-23).

Given the very specific format that the resultant constructed mask in Hao may assume, Hao's mask cannot specify any combination of bits corresponding to the positions in the destination register that should be loaded with the source operand's value. For example, Hao's mask cannot load the source operand into bytes 1 and 3 of the destination register since to so, the mask would have to be constructed as: "11111111000000001111111100000000". Such a mask cannot be constructed using the mask field in any of Hao's instructions since it includes two strings of consecutive 1's and two strings of consecutive 0's, which Hao's instruction mask field is incapable of producing.

Since Hao's mask cannot specify any combination of byte positions in a destination register, Hao therefore does not disclose or suggest "causing the ALU selectively load any specified combination of bytes of data within a transfer register associated with one of the

plurality of microengines with a shifted value of an operand that preserves the bytes of data that are not loaded", as required by applicant's independent claim 1.

Hennessey's "Computer organization and Design" generally describes the benefits of connecting multiple processors together (see page 712). Hennessy does not provide any technical details of any sort regarding the configuration and operation of such interconnected multiple processor. Accordingly, Hennessey's "Computer organization and Design" does not disclose or suggest instructions or operations for "causing the ALU to selectively load any specified combination of bytes of data within a transfer register associated with one of the plurality of microengines with a shifted value of an operand that preserves the bytes of data that are not loaded", as required by applicant's independent claim 1.

Hennessy's "Computer Architecture A Quantitative Approach" describes a protection mechanism to protect processes' states during context switches by restricting different processes to operate within memory bounds (see page 448). Nowhere does Hennessy's "Computer Architecture A Quantitative Approach" describe instructions or operations for "causing the ALU to selectively load any specified combination of bytes of data within a transfer register associated with one of the plurality of microengines with a shifted value of an operand that preserves the bytes of data that are not loaded", as required by applicant's independent claim 1.

Since none of the cited references discloses or suggests, alone or in combination, an instruction that causes the ALU to "selectively load any specified combination of bytes of data within a transfer register associated with one of the plurality of microengines with a shifted value of an operand that preserves the bytes of data that are not loaded", claim 1 is patentable over the cited references. Claims 2-9 depend from claim 1 and are therefore patentable for at least the same reasons that claim 1 is patentable.

Independent claims 10 and 19 recite the feature "loading selectively any combination of bytes of data within a register associated with one of a plurality of microengines with a shifted value of an operand", or similar language. At least this feature is not disclosed by the cited art. Accordingly, independent claims 10 and 19 are patentable over the cited art. Claims 11-18 depend from independent claim 10 and are therefore patentable for at least the same reasons as

claim 10. Claims 20-27 depend from independent claim 19 and are therefore patentable for at least the same reasons as independent claim 19.

Additionally, as noted, the examiner rejected claims 9, 18, and 27 under 35 U.S.C. §103(a) as being unpatentable over Hao in view of "Computer Organization and Design" to Hennessy et al., and further in view of Kiuchi. The examiner admitted that Hao and "Computer Organization and Design" do not teach an optional token set by the programmer and specifies to set ALU condition codes based on the result (page 11, paragraph 36 of the July 9, 2005, Office Action). The examiner, however, argued that Kiuchi describes this feature. Applicant respectfully disagrees.

Kiuchi describes a digital signal processor (Abstract). While Kiuchi discloses an instruction word that also includes a condition code field that identifies a predefined condition and also identifies whether the condition code register should be updated when the data processing operation is performed by the execution unit (abstract), nowhere does Kiuchi disclose or suggest that such a condition code specifies loading ALU condition codes, and certainly it does not disclose or suggest doing so based on a result formed in the ALU, as required by applicant's amended claim 9.

Since none of the reference cited by the examiner discloses or suggests, alone or in combination, the feature of "an optional token that is set by a programmer and specifies to set arithmetic logic unit (ALU) condition codes based on a result formed in the ALU," applicant's claim 9 is thus patentable over the cited art.

Claims 18 and 27 recite "an optional token that is set by a programmer and specifies to load arithmetic logic unit (ALU) condition codes based on a result formed in the ALU", or similar language. For similar reasons as those provided with respect to claim 9, at least this feature is not disclosed or suggested by the cited art. Accordingly, applicant's claims 18 and 27 are also patentable over the cited art.

It is believed that all the rejections and/or objections raised by the examiner have been addressed.

Canceled claims, if any, have been canceled without prejudice or disclaimer.

Any circumstance in which the applicant has (a) addressed certain comments of the examiner does not mean that the applicant concedes other comments of the examiner, (b) made arguments for the patentability of some claims does not mean that there are not other good reasons for patentability of those claims and other claims, or (c) amended or canceled a claim does not mean that the applicant concedes any of the examiner's positions with respect to that claim or other claims.

Please apply any charges to deposit account 06-1050, referencing attorney docket 10559-309US1.

Respectfully submitted,

Date: Sept. 7, 2005

Ido Rabinovitch
Reg. No. L0080

PTO Customer No. 26161
Fish & Richardson P.C.
225 Franklin Street
Boston, MA 02110-2804
Telephone: (617) 542-5070
Facsimile: (617) 542-8906